MUHAMMAD HAMMAD RAZA

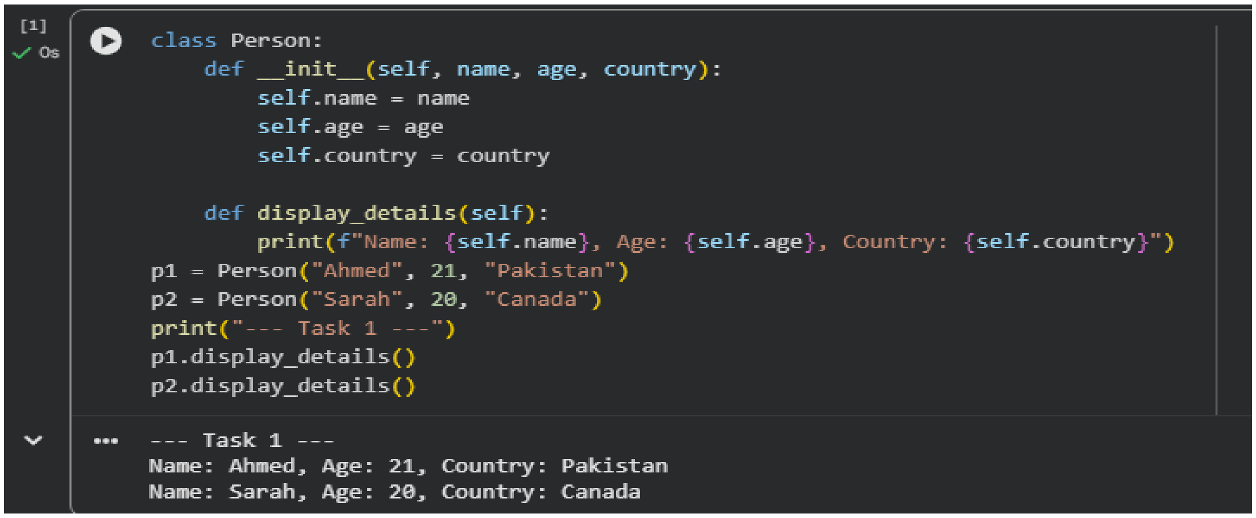
BATCH#10

ENROLLMENT: 512343

SMIT

MALIR CAMPUS, KARACHI

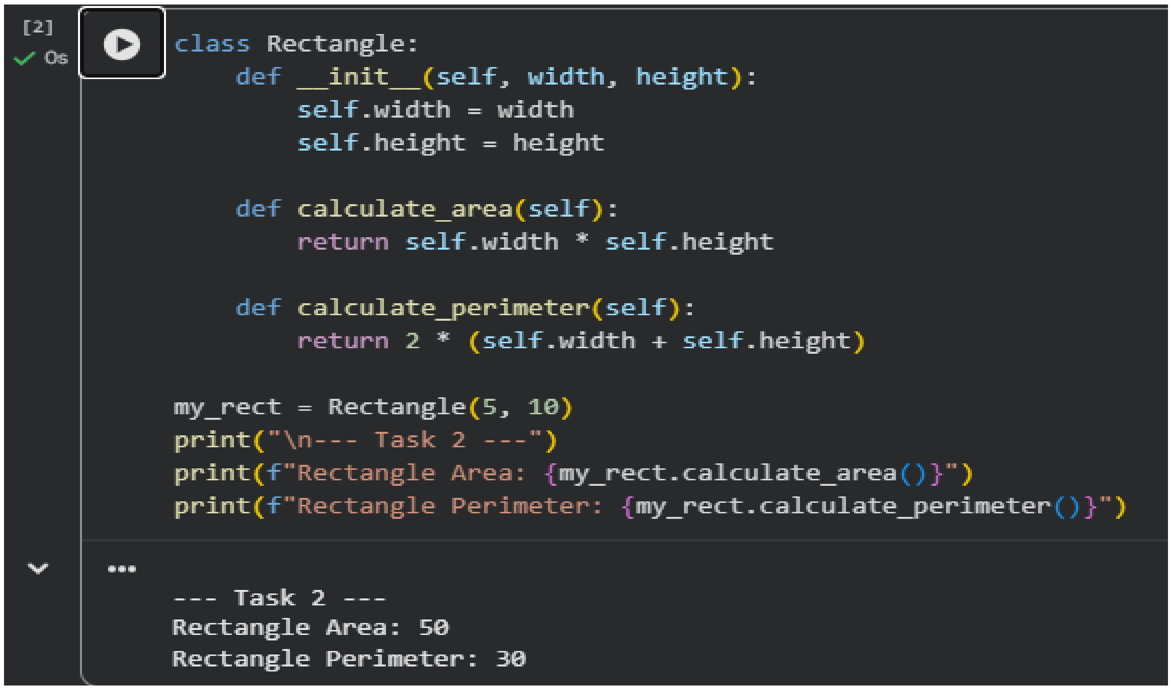
Program 01:



```
class Person:
    def __init__(self, name, age, country):
        self.name = name
        self.age = age
        self.country = country

    def display_details(self):
        print(f"Name: {self.name}, Age: {self.age}, Country: {self.country}")
p1 = Person("Ahmed", 21, "Pakistan")
p2 = Person("Sarah", 20, "Canada")
print("--- Task 1 ---")
p1.display_details()
p2.display_details()
```

```
--- Task 1 ---
Name: Ahmed, Age: 21, Country: Pakistan
Name: Sarah, Age: 20, Country: Canada
```

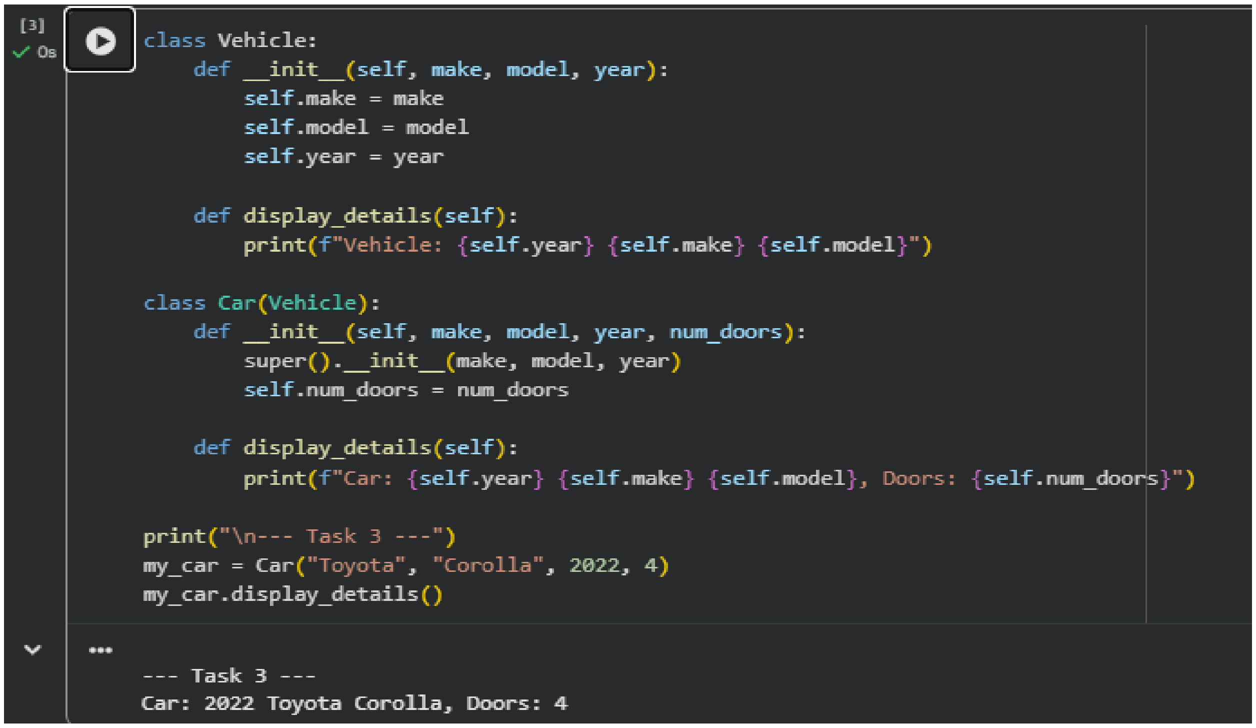
Program 02:



```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def calculate_area(self):
        return self.width * self.height

    def calculate_perimeter(self):
        return 2 * (self.width + self.height)

my_rect = Rectangle(5, 10)
print("\n--- Task 2 ---")
print(f"Rectangle Area: {my_rect.calculate_area()}")
print(f"Rectangle Perimeter: {my_rect.calculate_perimeter()}")
```

```

--- Task 2 ---
Rectangle Area: 50
Rectangle Perimeter: 30
```

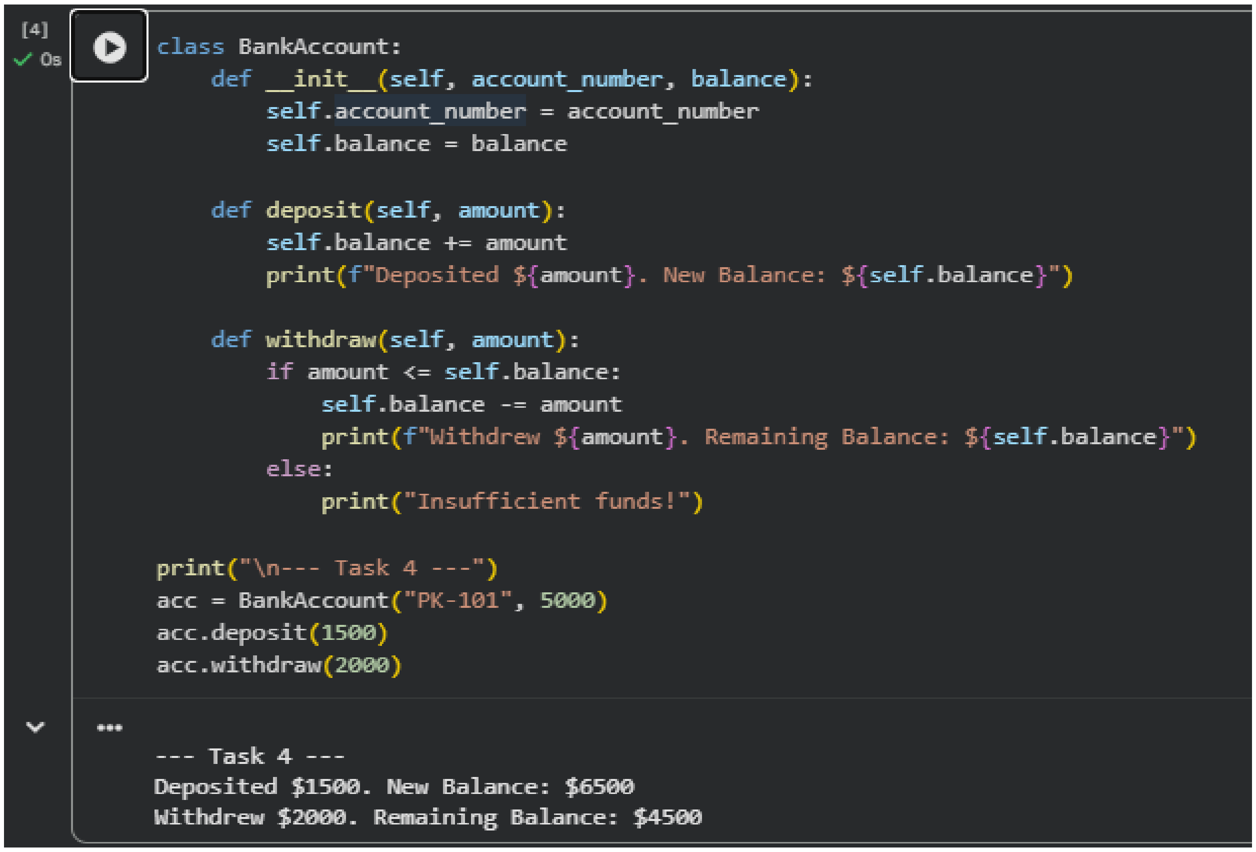
Program 03:



```
class Vehicle:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display_details(self):
        print(f"Vehicle: {self.year} {self.make} {self.model}")

class Car(Vehicle):
    def __init__(self, make, model, year, num_doors):
        super().__init__(make, model, year)
        self.num_doors = num_doors

    def display_details(self):
        print(f"Car: {self.year} {self.make} {self.model}, Doors: {self.num_doors}")

print("\n--- Task 3 ---")
my_car = Car("Toyota", "Corolla", 2022, 4)
my_car.display_details()
```

```
--- Task 3 ---
Car: 2022 Toyota Corolla, Doors: 4
```

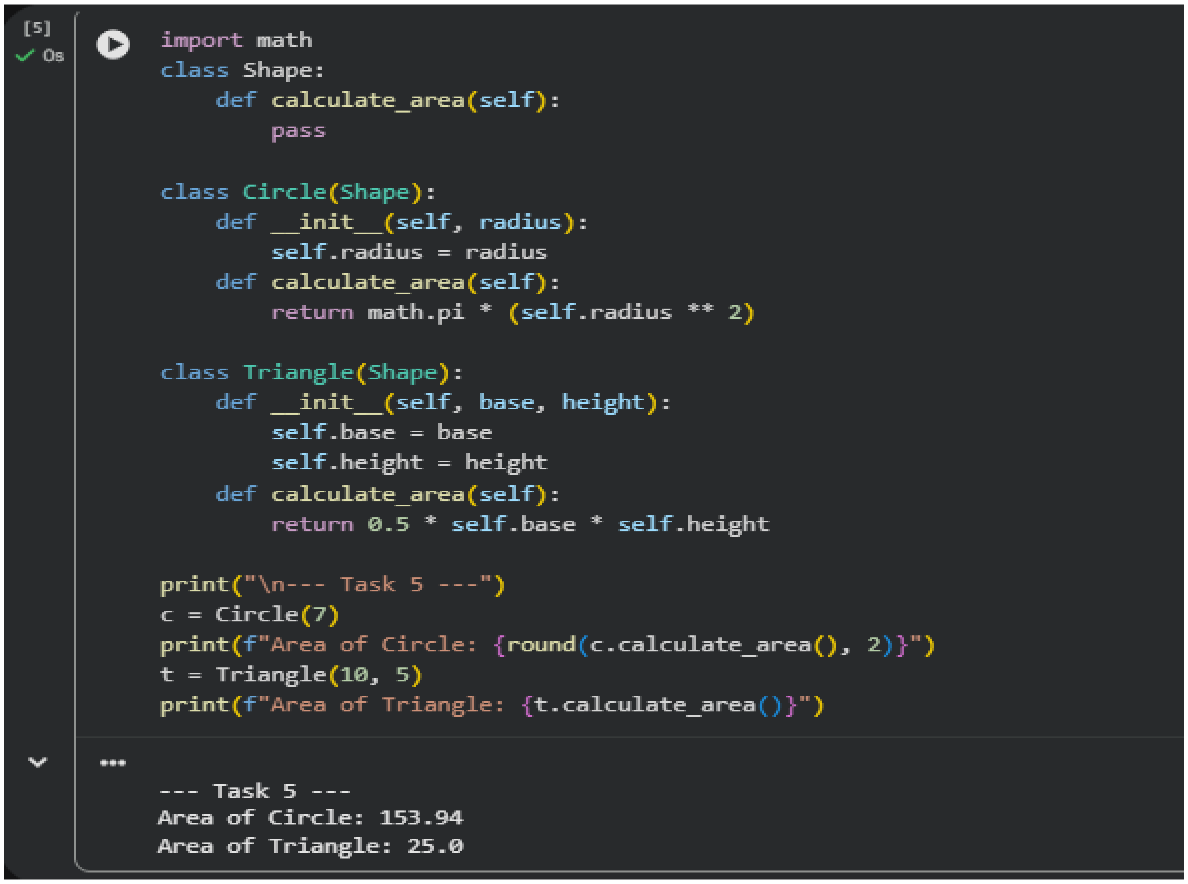
Program 04:



```
class BankAccount:
    def __init__(self, account_number, balance):
        self.account_number = account_number
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited ${amount}. New Balance: ${self.balance}")

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew ${amount}. Remaining Balance: ${self.balance}")
        else:
            print("Insufficient funds!")

print("\n--- Task 4 ---")
acc = BankAccount("PK-101", 5000)
acc.deposit(1500)
acc.withdraw(2000)
```

```
--- Task 4 ---
Deposited $1500. New Balance: $6500
Withdrew $2000. Remaining Balance: $4500
```

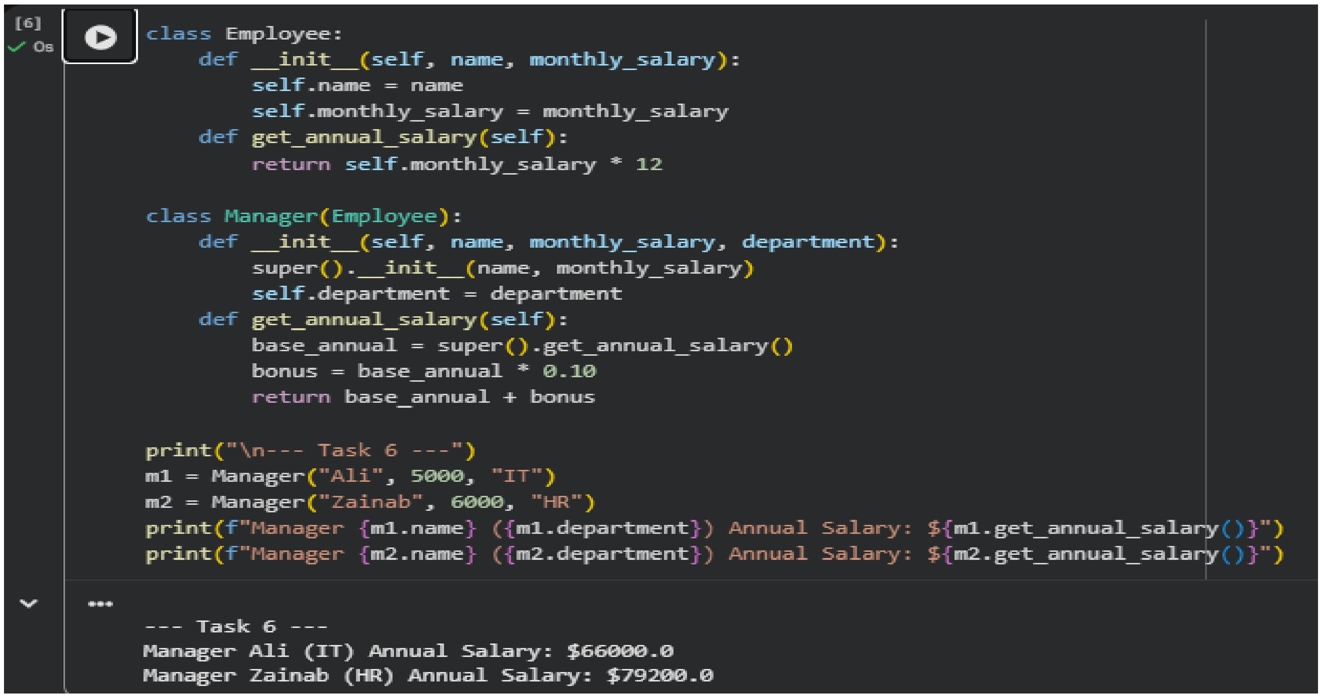
Program 05:



```
[5]
✓ 0s
import math
class Shape:
    def calculate_area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def calculate_area(self):
        return math.pi * (self.radius ** 2)

class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height
    def calculate_area(self):
        return 0.5 * self.base * self.height

print("\n--- Task 5 ---")
c = Circle(7)
print(f"Area of Circle: {round(c.calculate_area(), 2)}")
t = Triangle(10, 5)
print(f"Area of Triangle: {t.calculate_area()}")
```

```
--- Task 5 ---
Area of Circle: 153.94
Area of Triangle: 25.0
```

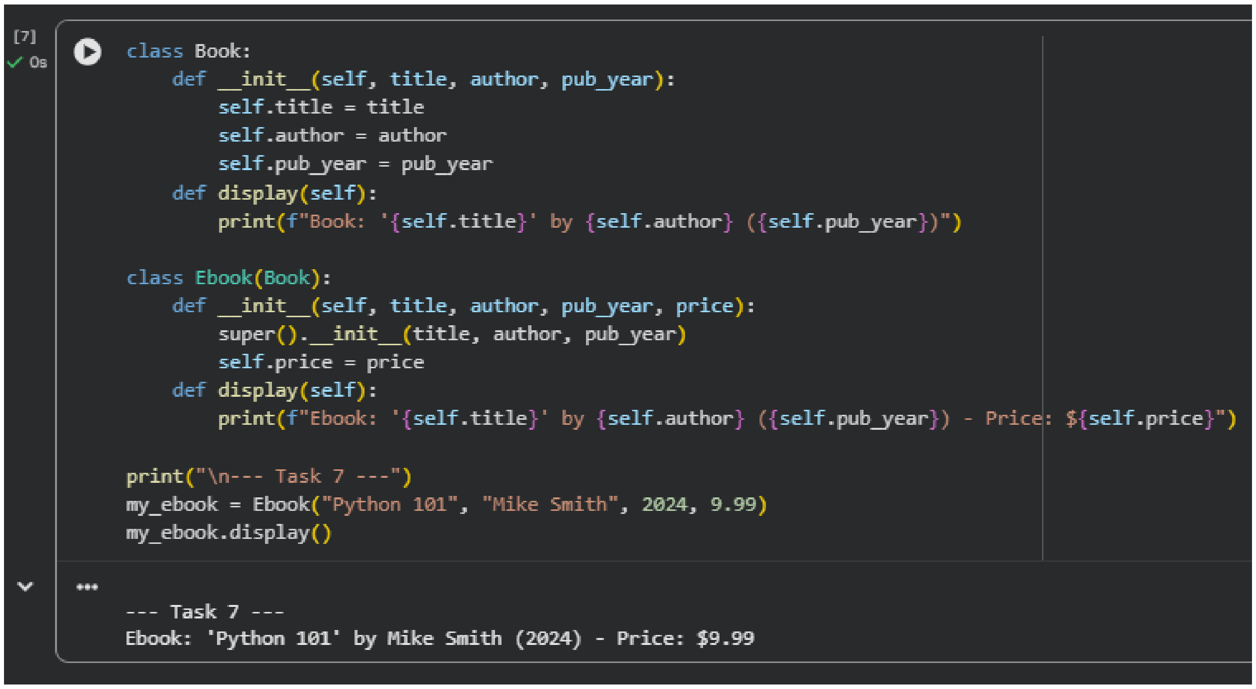
Program 06:



```
[6]
✓ 0s
class Employee:
    def __init__(self, name, monthly_salary):
        self.name = name
        self.monthly_salary = monthly_salary
    def get_annual_salary(self):
        return self.monthly_salary * 12

class Manager(Employee):
    def __init__(self, name, monthly_salary, department):
        super().__init__(name, monthly_salary)
        self.department = department
    def get_annual_salary(self):
        base_annual = super().get_annual_salary()
        bonus = base_annual * 0.10
        return base_annual + bonus

print("\n--- Task 6 ---")
m1 = Manager("Ali", 5000, "IT")
m2 = Manager("Zainab", 6000, "HR")
print(f"Manager {m1.name} ({m1.department}) Annual Salary: ${m1.get_annual_salary()}")
print(f"Manager {m2.name} ({m2.department}) Annual Salary: ${m2.get_annual_salary()}")
```

```
--- Task 6 ---
Manager Ali (IT) Annual Salary: $66000.0
Manager Zainab (HR) Annual Salary: $79200.0
```

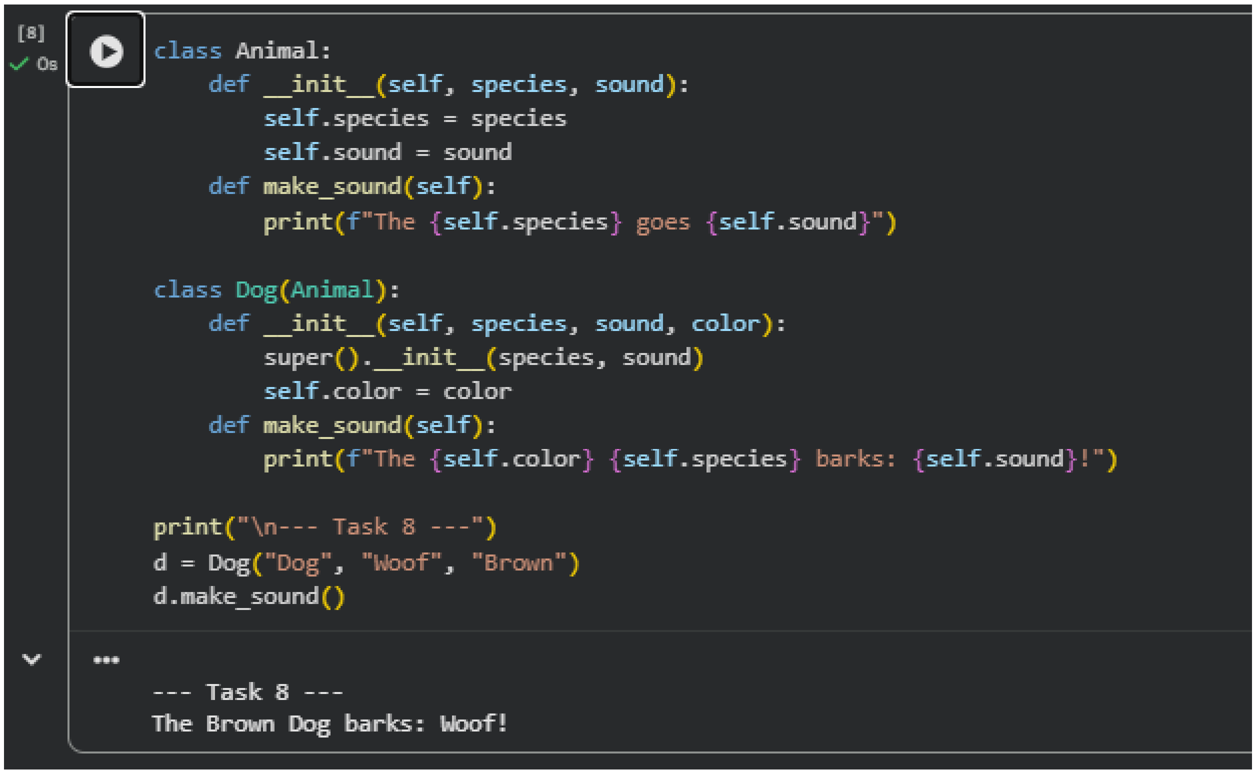
Program 07:



```
[7]
class Book:
    def __init__(self, title, author, pub_year):
        self.title = title
        self.author = author
        self.pub_year = pub_year
    def display(self):
        print(f"Book: '{self.title}' by {self.author} ({self.pub_year})")

class Ebook(Book):
    def __init__(self, title, author, pub_year, price):
        super().__init__(title, author, pub_year)
        self.price = price
    def display(self):
        print(f"Ebook: '{self.title}' by {self.author} ({self.pub_year}) - Price: ${self.price}")

print("\n--- Task 7 ---")
my_ebook = Ebook("Python 101", "Mike Smith", 2024, 9.99)
my_ebook.display()
```

```
--- Task 7 ---
Ebook: 'Python 101' by Mike Smith (2024) - Price: $9.99
```

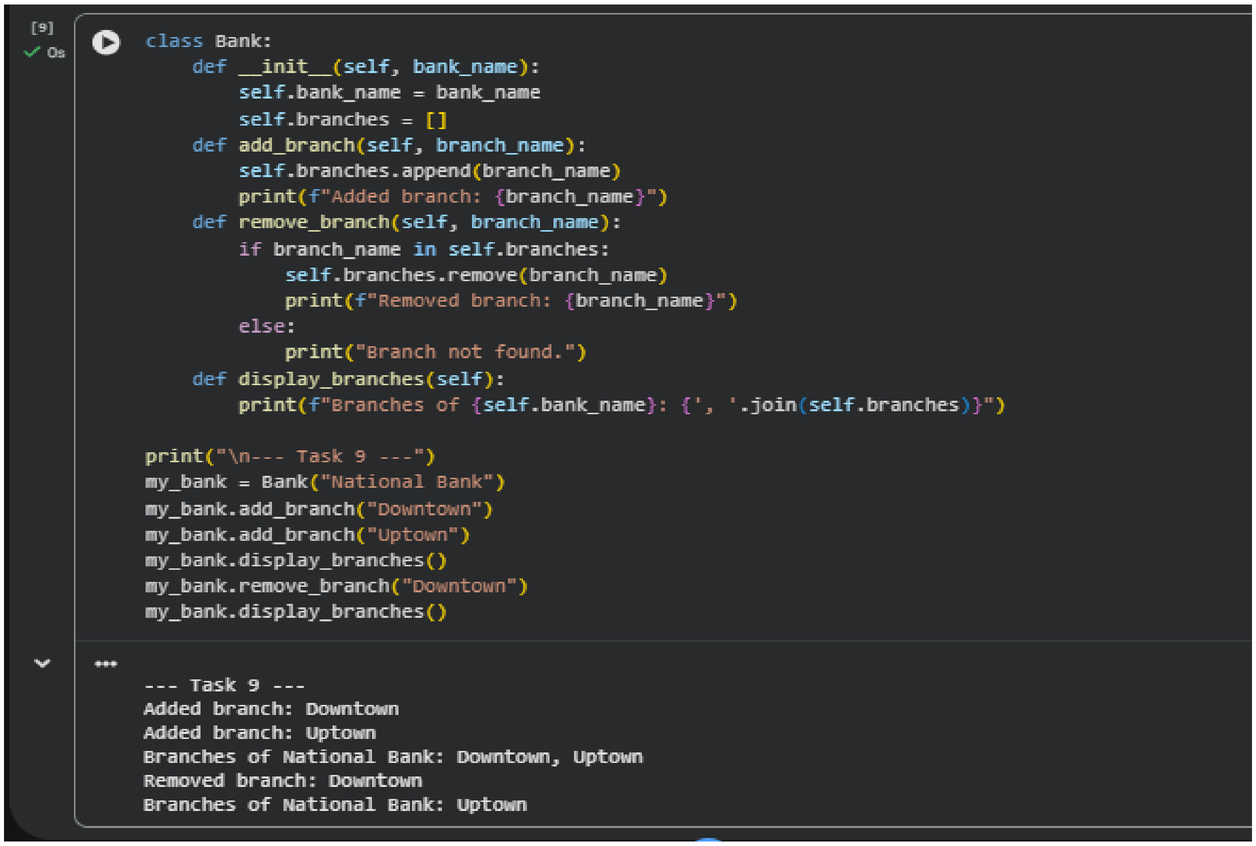
Program 08:



```
[8]
class Animal:
    def __init__(self, species, sound):
        self.species = species
        self.sound = sound
    def make_sound(self):
        print(f"The {self.species} goes {self.sound}")

class Dog(Animal):
    def __init__(self, species, sound, color):
        super().__init__(species, sound)
        self.color = color
    def make_sound(self):
        print(f"The {self.color} {self.species} barks: {self.sound}!")

print("\n--- Task 8 ---")
d = Dog("Dog", "Woof", "Brown")
d.make_sound()
```

```
--- Task 8 ---
The Brown Dog barks: Woof!
```

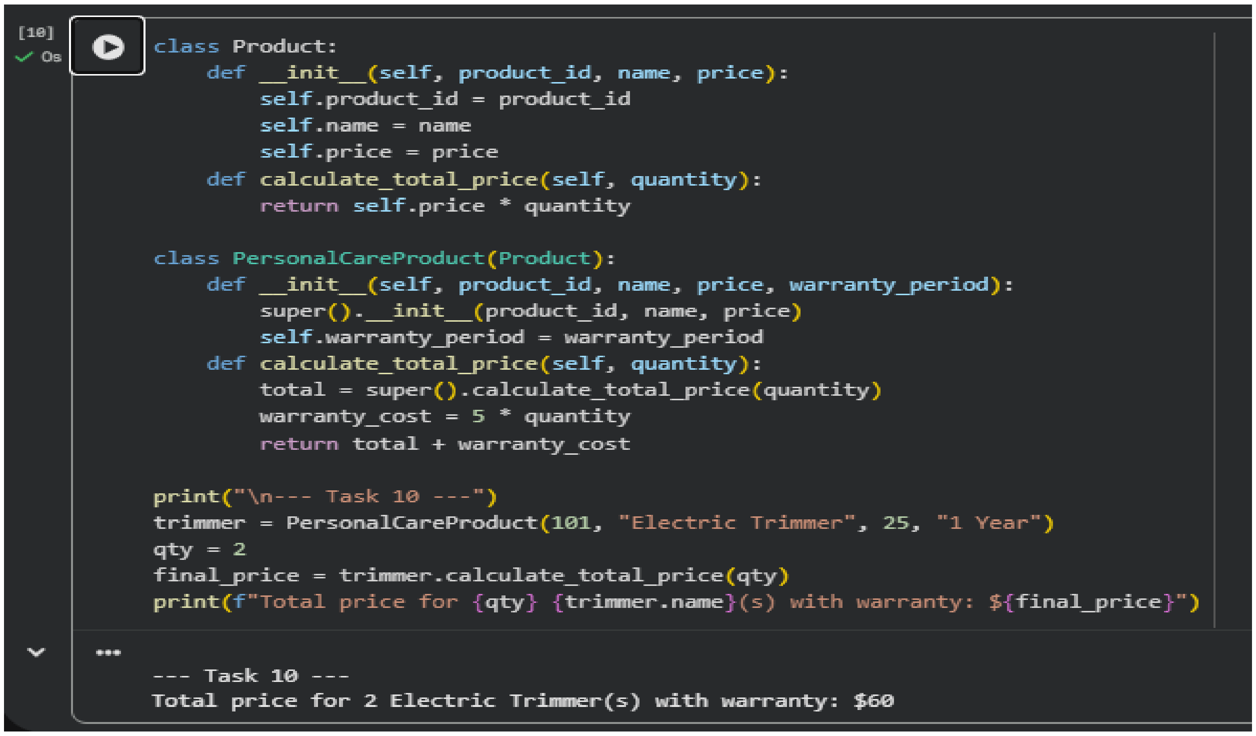
Program 09:



```
[9]
class Bank:
    def __init__(self, bank_name):
        self.bank_name = bank_name
        self.branches = []
    def add_branch(self, branch_name):
        self.branches.append(branch_name)
        print(f"Added branch: {branch_name}")
    def remove_branch(self, branch_name):
        if branch_name in self.branches:
            self.branches.remove(branch_name)
            print(f"Removed branch: {branch_name}")
        else:
            print("Branch not found.")
    def display_branches(self):
        print(f"Branches of {self.bank_name}: {', '.join(self.branches)}")

print("\n--- Task 9 ---")
my_bank = Bank("National Bank")
my_bank.add_branch("Downtown")
my_bank.add_branch("Uptown")
my_bank.display_branches()
my_bank.remove_branch("Downtown")
my_bank.display_branches()
```

```
--- Task 9 ---
Added branch: Downtown
Added branch: Uptown
Branches of National Bank: Downtown, Uptown
Removed branch: Downtown
Branches of National Bank: Uptown
```

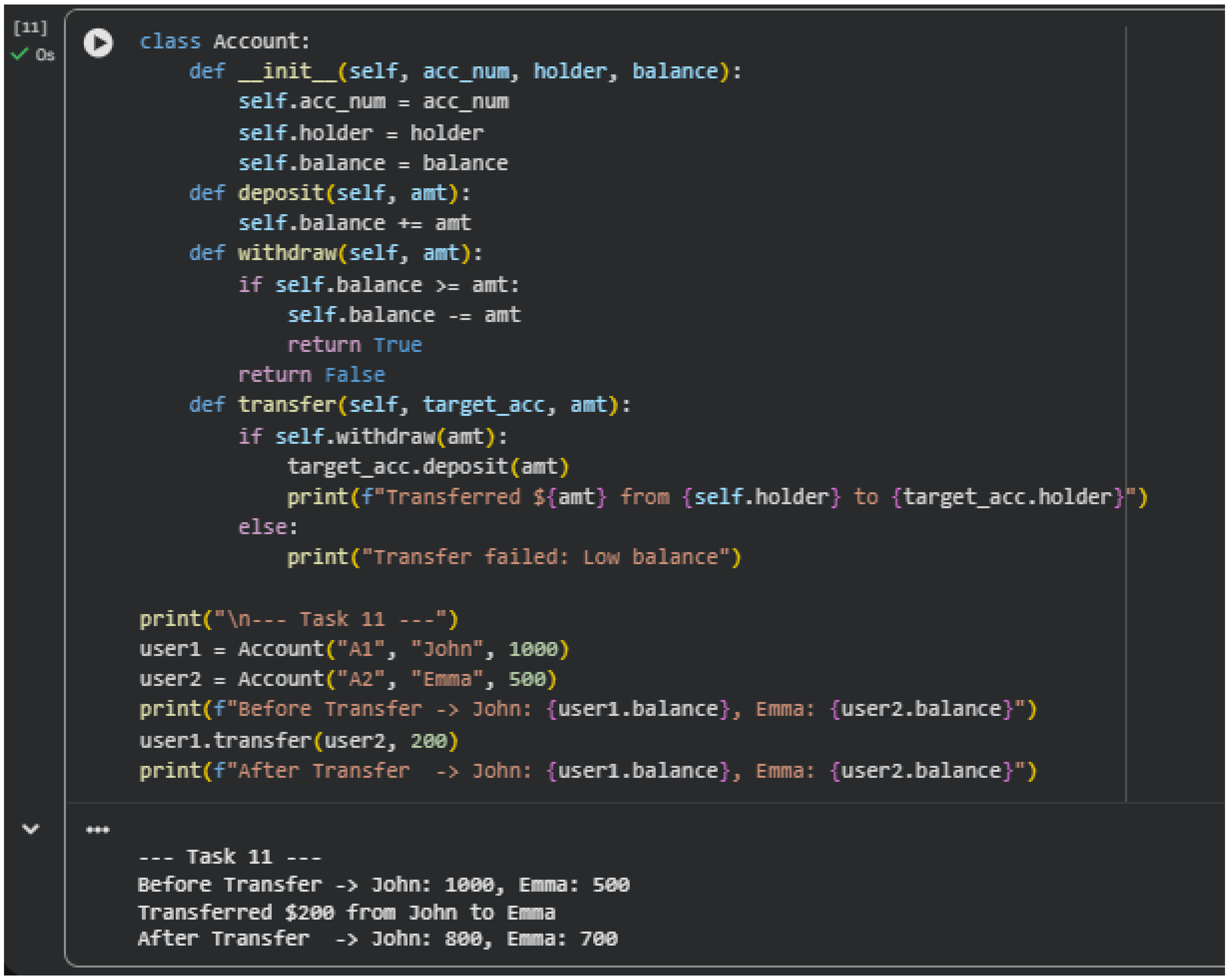
Program 10:



```
[10]
class Product:
    def __init__(self, product_id, name, price):
        self.product_id = product_id
        self.name = name
        self.price = price
    def calculate_total_price(self, quantity):
        return self.price * quantity

class PersonalCareProduct(Product):
    def __init__(self, product_id, name, price, warranty_period):
        super().__init__(product_id, name, price)
        self.warranty_period = warranty_period
    def calculate_total_price(self, quantity):
        total = super().calculate_total_price(quantity)
        warranty_cost = 5 * quantity
        return total + warranty_cost

print("\n--- Task 10 ---")
trimmer = PersonalCareProduct(101, "Electric Trimmer", 25, "1 Year")
qty = 2
final_price = trimmer.calculate_total_price(qty)
print(f"Total price for {qty} {trimmer.name}(s) with warranty: ${final_price}")
```

```
--- Task 10 ---
Total price for 2 Electric Trimmer(s) with warranty: $60
```

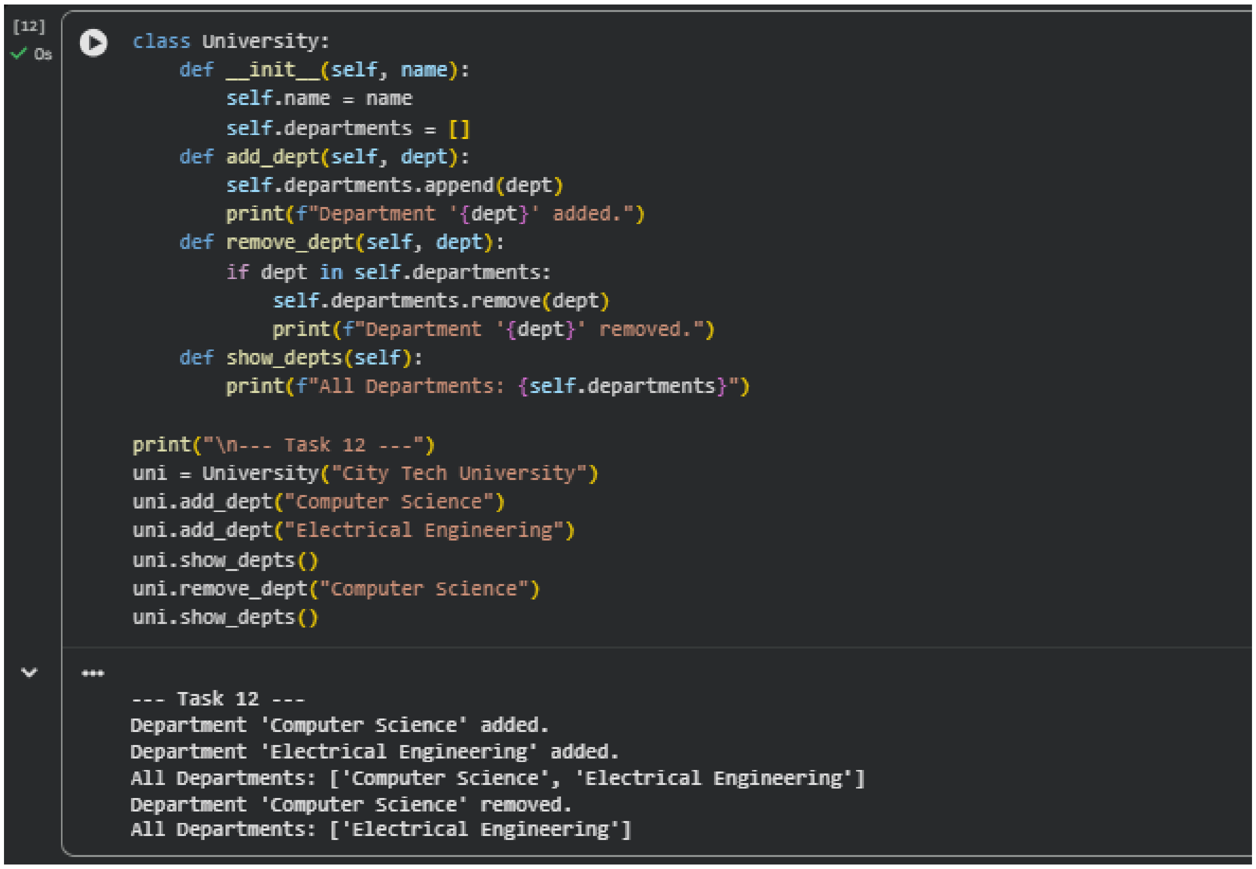
Program 11:



```
class Account:
    def __init__(self, acc_num, holder, balance):
        self.acc_num = acc_num
        self.holder = holder
        self.balance = balance
    def deposit(self, amt):
        self.balance += amt
    def withdraw(self, amt):
        if self.balance >= amt:
            self.balance -= amt
            return True
        return False
    def transfer(self, target_acc, amt):
        if self.withdraw(amt):
            target_acc.deposit(amt)
            print(f"Transferred ${amt} from {self.holder} to {target_acc.holder}")
        else:
            print("Transfer failed: Low balance")

print("\n--- Task 11 ---")
user1 = Account("A1", "John", 1000)
user2 = Account("A2", "Emma", 500)
print(f"Before Transfer -> John: {user1.balance}, Emma: {user2.balance}")
user1.transfer(user2, 200)
print(f"After Transfer  -> John: {user1.balance}, Emma: {user2.balance}")
```

```
--- Task 11 ---
Before Transfer -> John: 1000, Emma: 500
Transferred $200 from John to Emma
After Transfer  -> John: 800, Emma: 700
```

Program 12:



```
class University:
    def __init__(self, name):
        self.name = name
        self.departments = []
    def add_dept(self, dept):
        self.departments.append(dept)
        print(f"Department '{dept}' added.")
    def remove_dept(self, dept):
        if dept in self.departments:
            self.departments.remove(dept)
            print(f"Department '{dept}' removed.")
    def show_depts(self):
        print(f"All Departments: {self.departments}")

print("\n--- Task 12 ---")
uni = University("City Tech University")
uni.add_dept("Computer Science")
uni.add_dept("Electrical Engineering")
uni.show_depts()
uni.remove_dept("Computer Science")
uni.show_depts()
```

```
--- Task 12 ---
Department 'Computer Science' added.
Department 'Electrical Engineering' added.
All Departments: ['Computer Science', 'Electrical Engineering']
Department 'Computer Science' removed.
All Departments: ['Electrical Engineering']
```